Introduction to Reinforcement Learning based Control and Deep Reinforcement Learning based control

Associate Professor
(Maitre de Conférences)
Automatic Control, Reliability of Systems
mayank-shekhar.jha{at}univ-lorraine.fr

Research
Centre de Recherche en Automatique de Nancy
(CRAN) UMR 7039,
Faculté des Sciences et Technologies
Boulevard des Aiguillettes - BP 70239 - Bât.
1er cycle 54506 Vandoeuvre-lès-Nancy
Cedex France

Teaching
Polytech Nancy (ESSTIN),
2 Rue Jean Lamour 54509
Vandoeuvre-lès-Nancy,
Cedex France

Email: mayank-shekhar.jha [at] univ-lorraine.fr

Introduction

Markov Decision process

Dynamic Programming for planning and control

Dynamic Programming

Dynamic Programming

Policy Evaluation

Iterative approach required

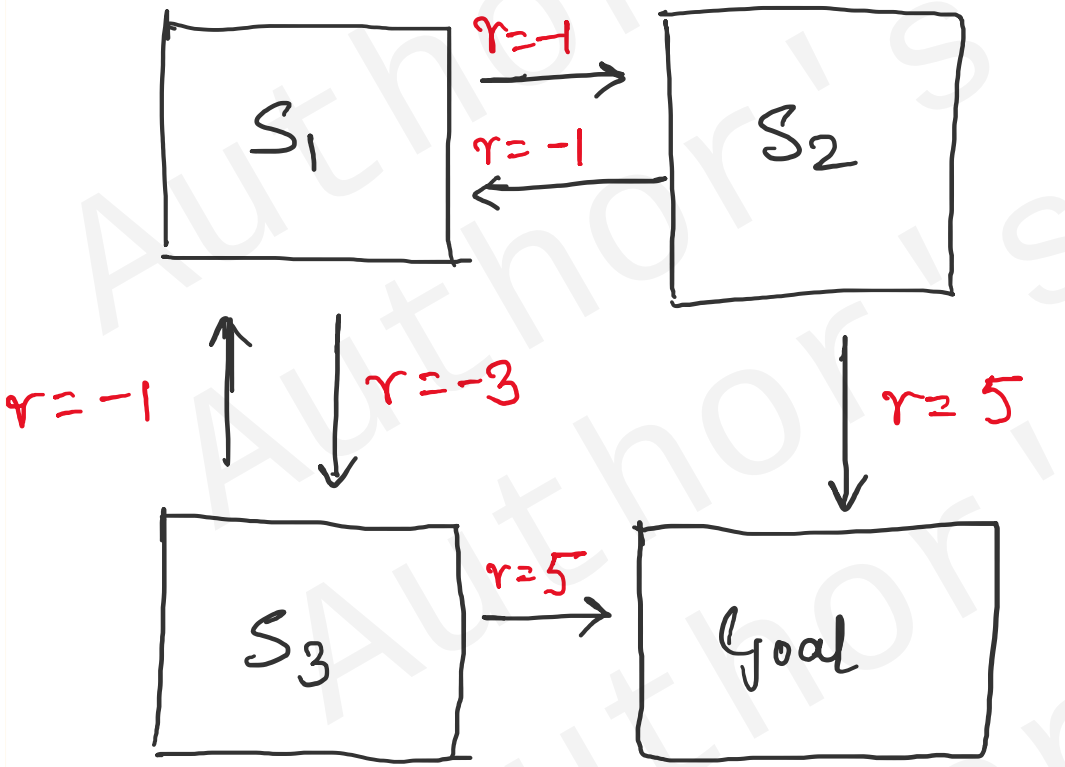
Dynamic Programming

Policy Evaluation

Iterative approach required

Motivation.

Grid World Example



Given π, determine $V_\pi(s)$ $\forall s \in S$ → for all

Policy Example: π

Actions are drawn from a uniform distribution.

$\pi(\text{right} \mid s_1) = 1/2$

$\pi(\text{down} \mid s_1) = 1/2$

$\pi(\text{left} \mid s_2) = 1/2$

$\pi(\text{down} \mid s_2) = 1/2$

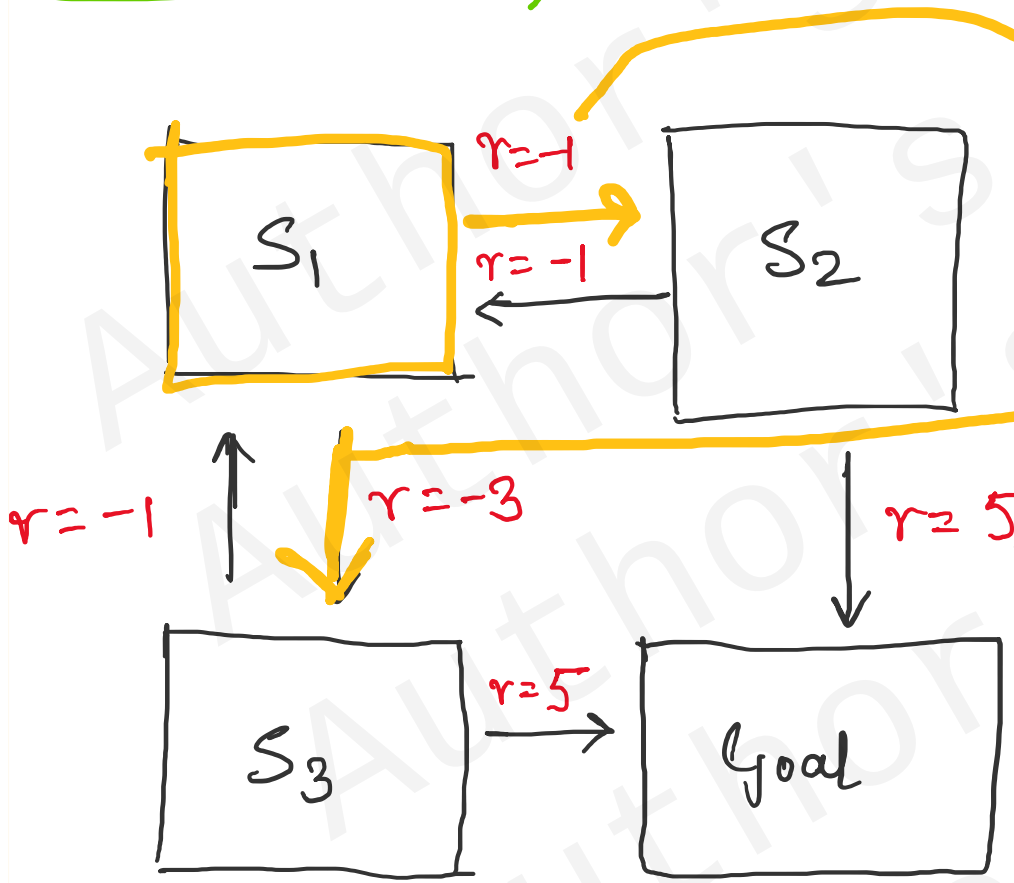
$\pi(\text{up} \mid s_2) = 1/2$

$\pi(\text{right} \mid s_2) = 1/2$

Actions are based on policy.

Policy depends on some probability

Grid World Example



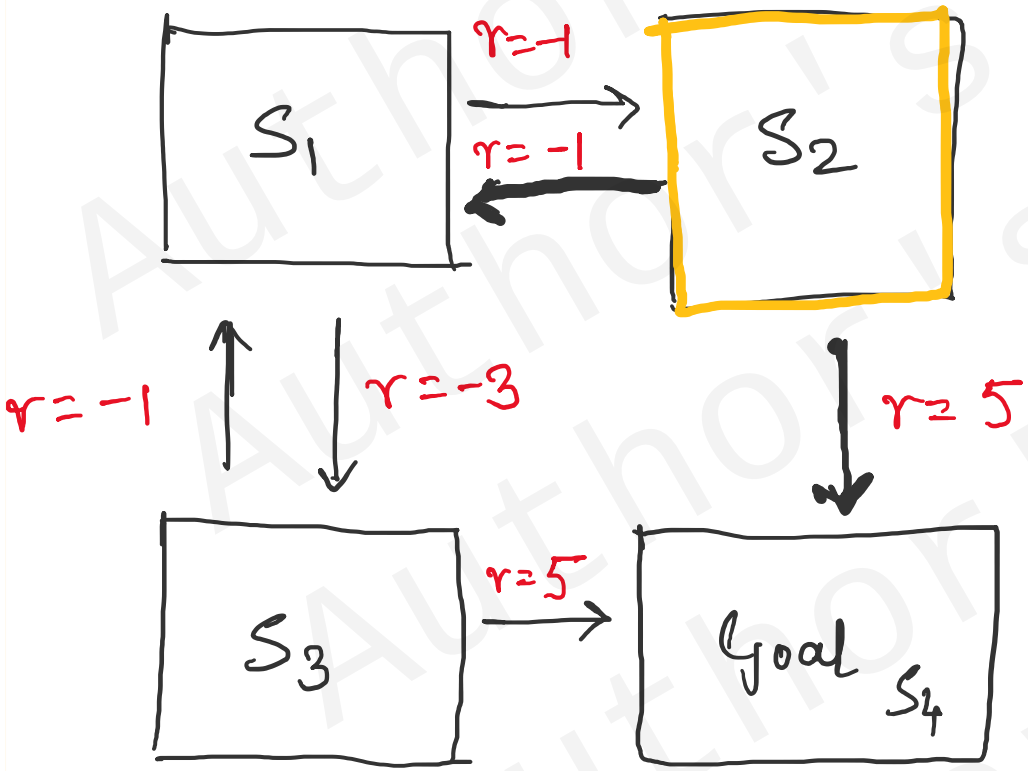
Half time

Other half time

$$V_\pi(s_1) = \frac{1}{2}\left(-1 + V_\pi(s_2)\right) + \frac{1}{2}\left(-3 + V_\pi(s_3)\right)$$

Given π, determine $V_\pi(s)$ $\forall s \in S$ → Random Uniform Policy.

Grid World Example



Given π, determine $V_\pi(s)$ $\forall s \in S$

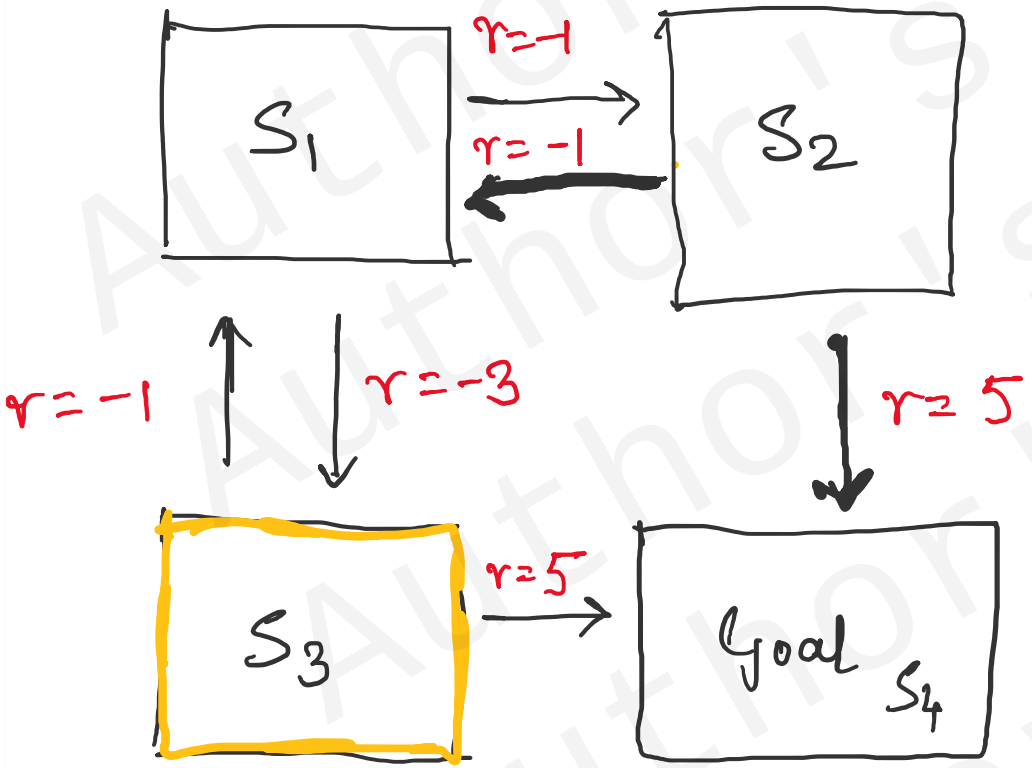
Half time

$$V_\pi(S_1) = \frac{1}{2}\left(-1 + V_\pi(S_2)\right) + \frac{1}{2}\left(-3 + V_\pi(S_3)\right)$$

Other half time

$$V_\pi(S_2) = \frac{1}{2}\left(-1 + V_\pi(S_1)\right) + \frac{1}{2}\left(5 + V_\pi(S_4)\right)$$

Grid World Example



Given π, determine $V_\pi(s)$ $\forall s \in S$

Half time

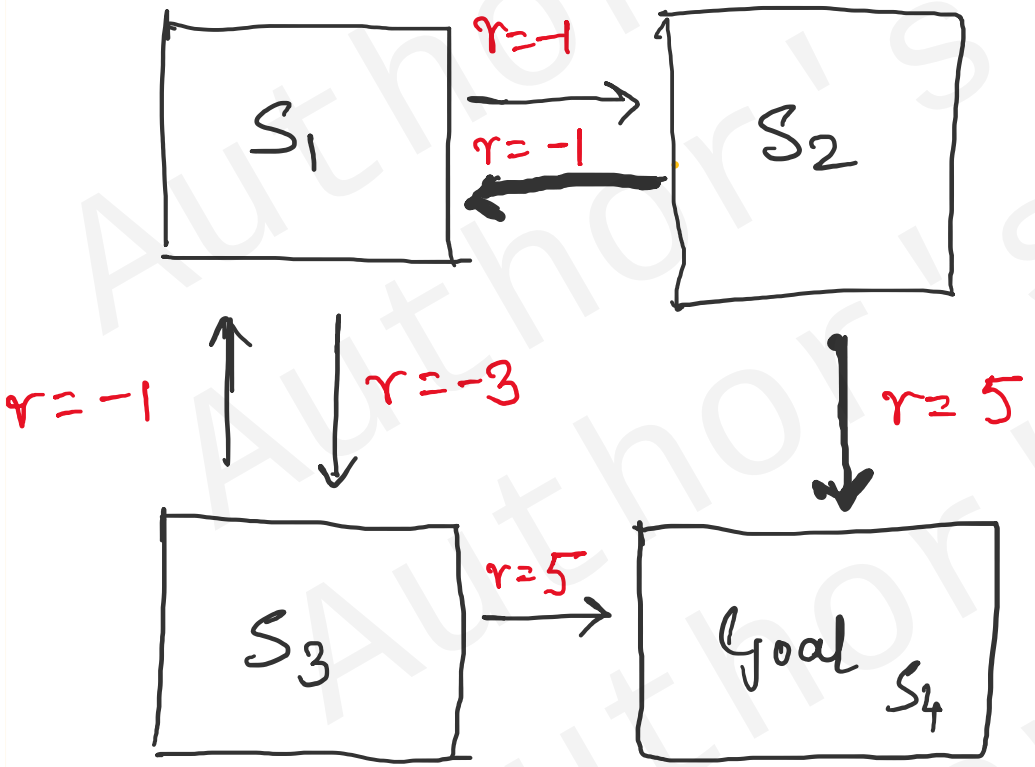
Other half time

$$V_\pi(S_1) = \frac{1}{2}\left(-1 + V_\pi(S_2)\right) + \frac{1}{2}\left(-3 + V_\pi(S_3)\right)$$

$$V_\pi(S_2) = \frac{1}{2}\left(-1 + V_\pi(S_1)\right) + \frac{1}{2}\left(5 + V_\pi(S_4)\right)$$

$$V_\pi(S_3) = \frac{1}{2}\left(-1 + V_\pi(S_1)\right) + \frac{1}{2}\left(5 + V_\pi(S_4)\right)$$

Grid World Example



Given π, determine $V_\pi(s)$ $\forall s \in S$

Half time

Other half time

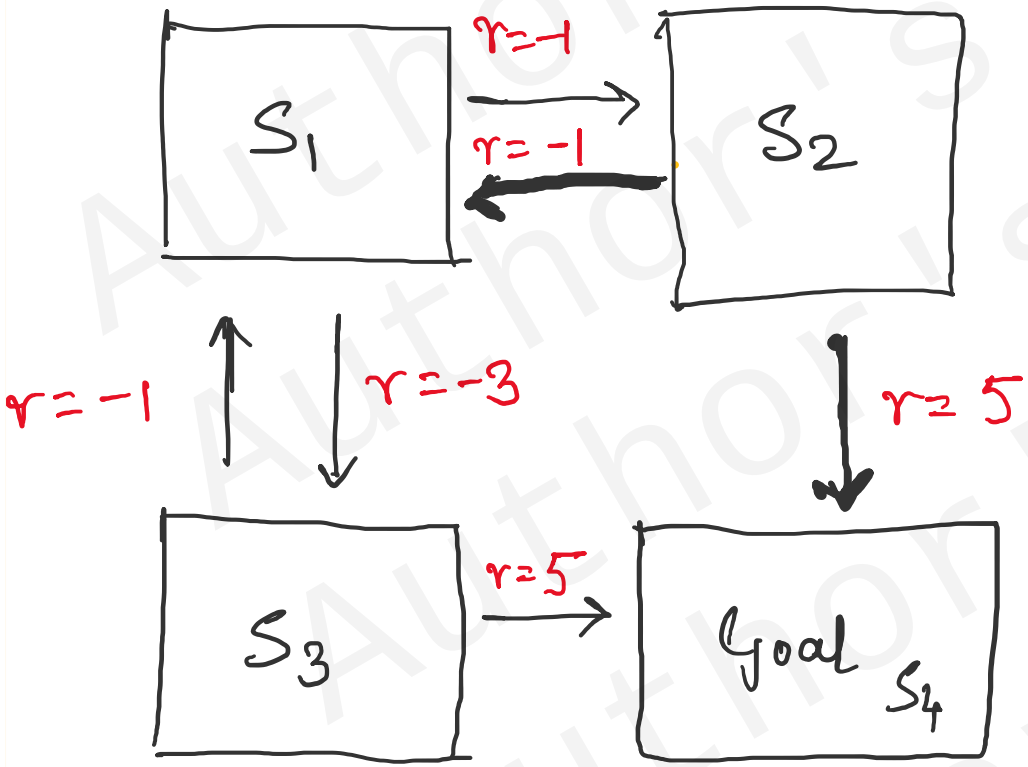
$$V_\pi(S_1) = \frac{1}{2}\left(-1 + V_\pi(S_2)\right) + \frac{1}{2}\left(-3 + V_\pi(S_3)\right)$$

$$V_\pi(S_2) = \frac{1}{2}\left(-1 + V_\pi(S_1)\right) + \frac{1}{2}\left(5 + V_\pi(S_4)\right)$$

$$V_\pi(S_3) = \frac{1}{2}\left(-1 + V_\pi(S_1)\right) + \frac{1}{2}\left(5 + V_\pi(S_4)\right)$$

$$V_\pi(S_4) = 0$$

Grid World Example



Given π, determine $V_\pi(s)$ $\forall s \in S$

$$V_\pi(s_1) = \frac{1}{2}(-1 + V_\pi(s_2)) + \frac{1}{2}(-3 + V_\pi(s_3))$$

$$V_\pi(s_2) = \frac{1}{2}(-1 + V_\pi(s_1)) + \frac{1}{2}(5 + V_\pi(s_4))$$

$$V_\pi(s_3) = \frac{1}{2}(-1 + V_\pi(s_1)) + \frac{1}{2}(5 + V_\pi(s_4))$$

$$V_\pi(s_4) = 0$$

4 equations, 4 unknown variables

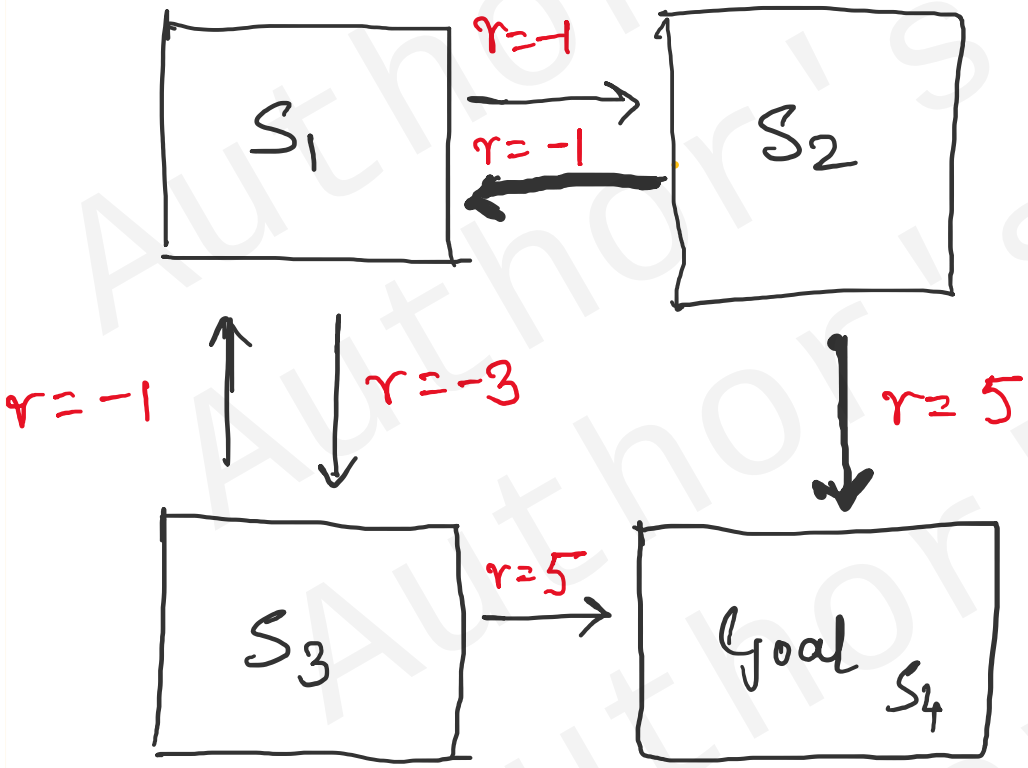
Can be solved to give

$V_\pi(s_1) = 0$, $V_\pi(s_4) = 0$

$V_\pi(s_2) = 2$

$V_\pi(s_3) = 2$

Grid World Example



Given π, determine $V_\pi(s)$ $\forall s \in S$

$$V_\pi(S_1) = \frac{1}{2}\left(-1 + V_\pi(S_2)\right) + \frac{1}{2}\left(-3 + V_\pi(S_3)\right)$$

$$V_\pi(S_2) = \frac{1}{2}\left(-1 + V_\pi(S_1)\right) + \frac{1}{2}\left(5 + V_\pi(S_4)\right)$$

$$V_\pi(S_3) = \frac{1}{2}\left(-1 + V_\pi(S_1)\right) + \frac{1}{2}\left(5 + V_\pi(S_4)\right)$$

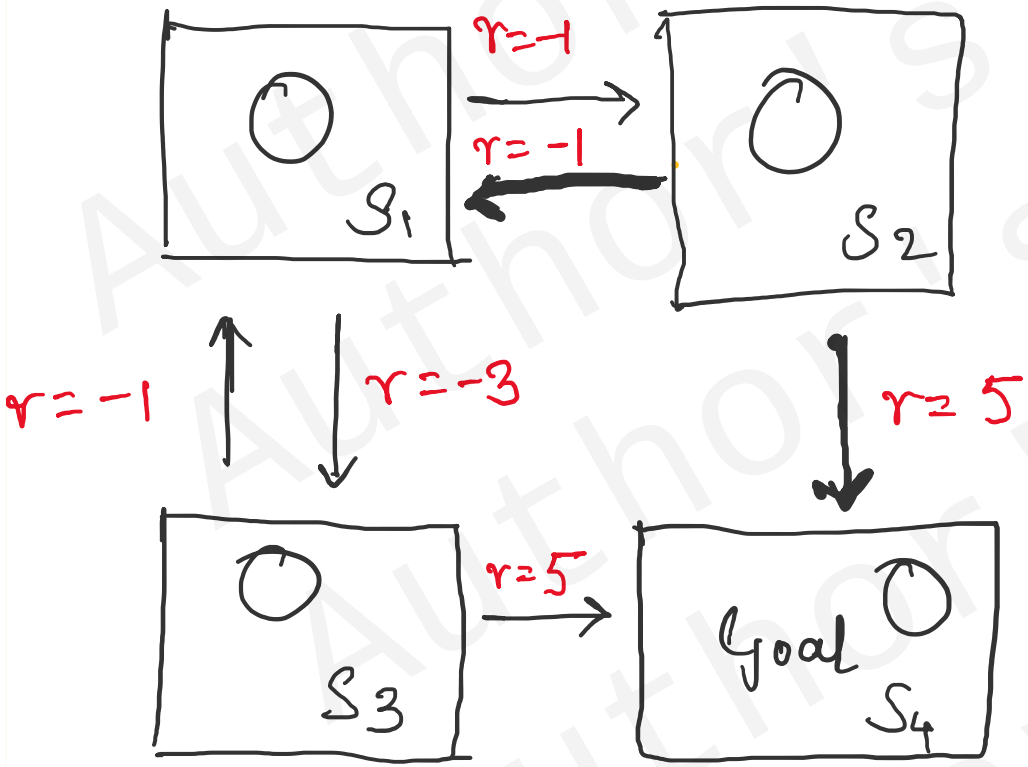
$$V_\pi(S_4) = 0$$

How to handle large state space ?

How to optimise under several policies / large states ??

Iterative Approach.

Grid World Example (Iterative Approach)



Given π, determine $V_\pi(s)$ $\forall s \in S$

→ start with a guess

$\{S_1=0, S_2=0, S_3=0, S_4=0\}$

→ Consider one state S_1

→ Improve the guess with respect to that state

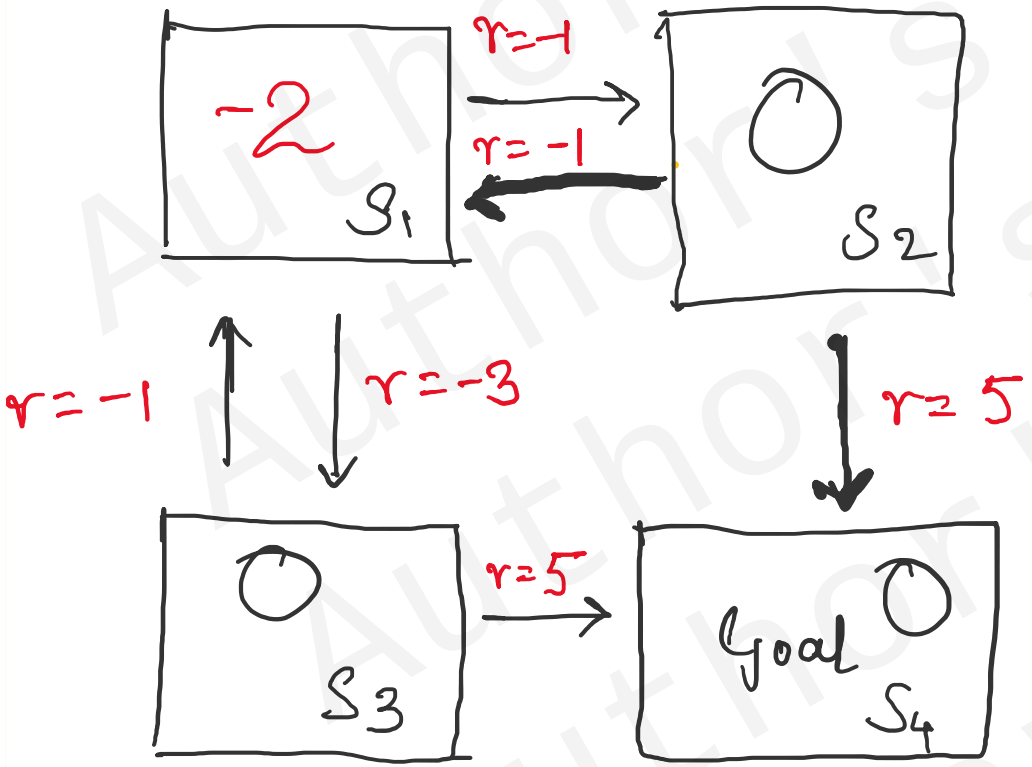
$$V_\pi(S_1) = \frac{1}{2}(-1 + V_\pi(S_2)) + \frac{1}{2}(-3 + V_\pi(S_3))$$

Use this to update the guess $V_\pi(S_1)$

update guess with a guess

$$V_\pi(S_1) \leftarrow \frac{1}{2}(-1+0) + \frac{1}{2}(-3+0) = -2$$

Grid World Example (Iterative Approach)

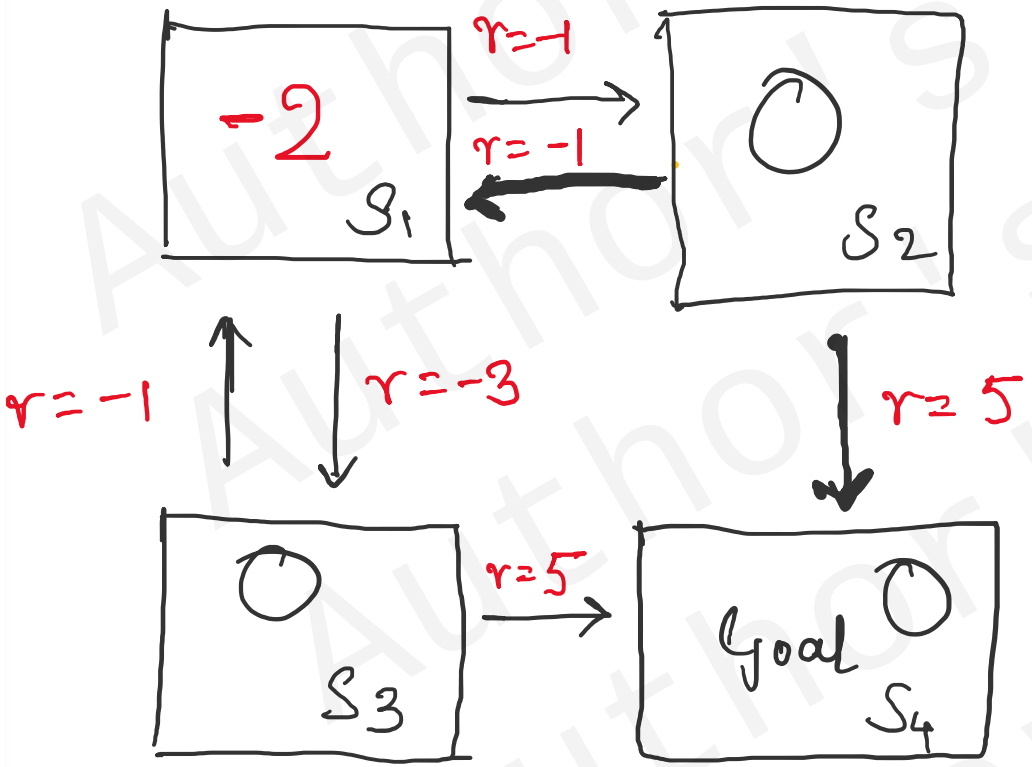


Use $V_\pi(s_1)$ to update value of s_2

$$V_\pi(s_2) \leftarrow \frac{1}{2}(-1 + V_\pi(s_1)) + \frac{1}{2}(5)$$

Given π, determine $V_\pi(s)$ $\forall s \in S$

Grid World Example (Iterative Approach)



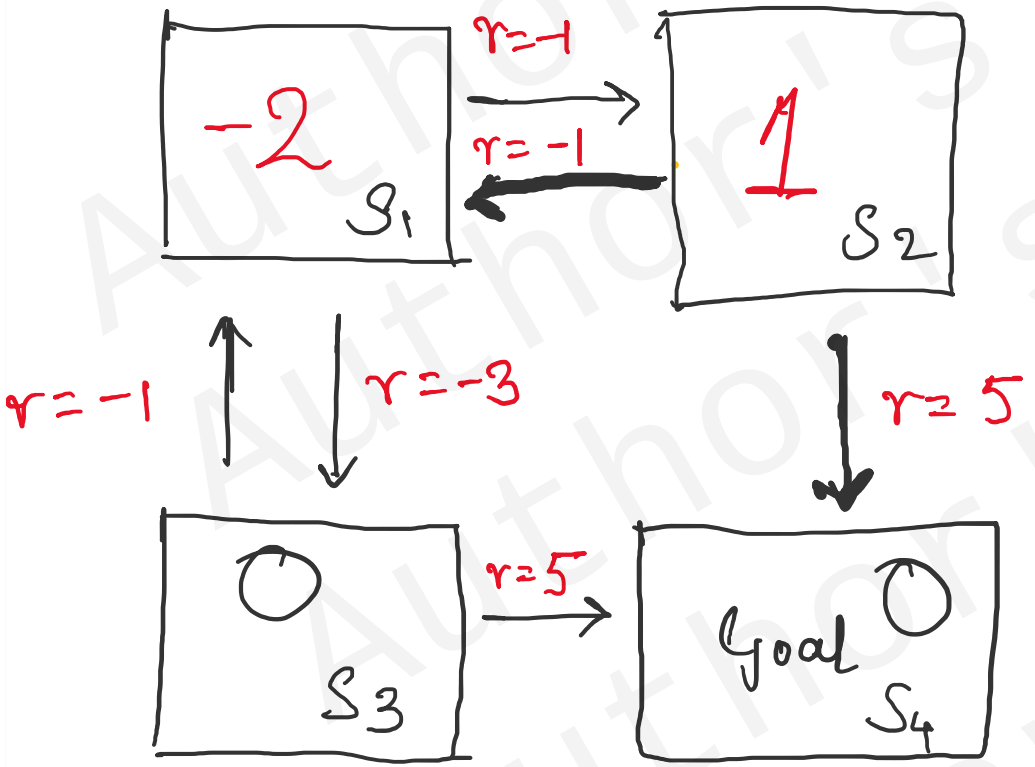
Use $V_\pi(s_1)$ to update value of s_2

$$V_\pi(s_2) \leftarrow \frac{1}{2}(-1 + V_\pi(s_1)) + \frac{1}{2}(5)$$

$$V_\pi(s_2) \leftarrow 1$$

Given π, determine $V_\pi(s)$ $\forall s \in S$

Grid World Example (Iterative Approach)



Given π, determine $V_\pi(s)$ $\forall s \in S$

Use $V_\pi(s_1)$ to update value of s_2

$$V_\pi(s_2) \leftarrow \tfrac{1}{2}(-1 + V_\pi(s_1)) + \tfrac{1}{2}(5)$$

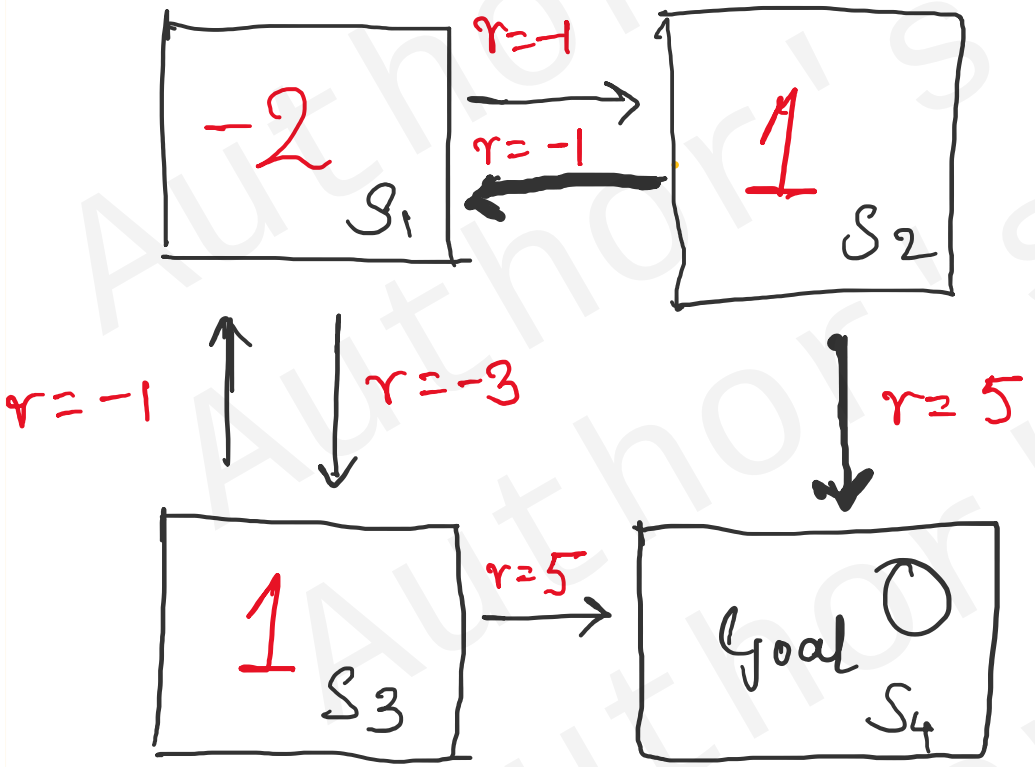
$V_\pi(s_2) \leftarrow 1$ → record

keep track and update $V_\pi(s_3)$

$$V_\pi(s_3) \leftarrow \tfrac{1}{2}(-1 + V_\pi(s_1)) + \tfrac{1}{2}(5)$$

$$= \tfrac{1}{2}(-1 + -2) + \tfrac{1}{2}(5) = 1$$

Grid World Example (Iterative Approach)



Given π, determine $V_\pi(s)$ $\forall s \in S$

Use $V_\pi(s_1)$ to update value of s_2

$$V_\pi(s_2) \leftarrow \tfrac{1}{2}(-1 + V_\pi(s_1)) + \tfrac{1}{2}(5)$$

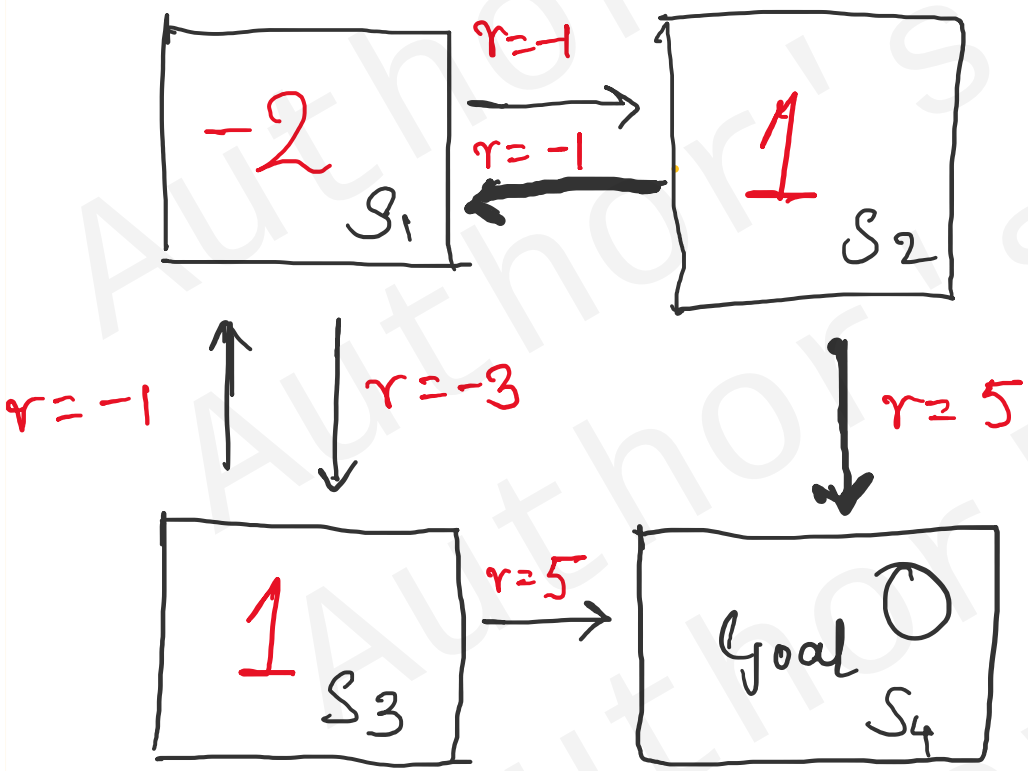
$V_\pi(s_2) \leftarrow \underline{1}$ → record

keep track and update $V_\pi(s_3)$

$$V_\pi(s_3) \leftarrow \tfrac{1}{2}(-1 + V_\pi(s_1)) + \tfrac{1}{2}(5)$$

$$= \tfrac{1}{2}(-1 + -2) + \tfrac{1}{2}(5) = \underline{1}$$

Grid World Example (Iterative Approach)



Given π, determine $V_\pi(s)$ $\forall s \in S$

→ Repeat this procedure

→ Visit the states over and over

→ Converge to True values of states !

Idea behind → Iterative Policy Evaluation

Iterative Policy Evaluation

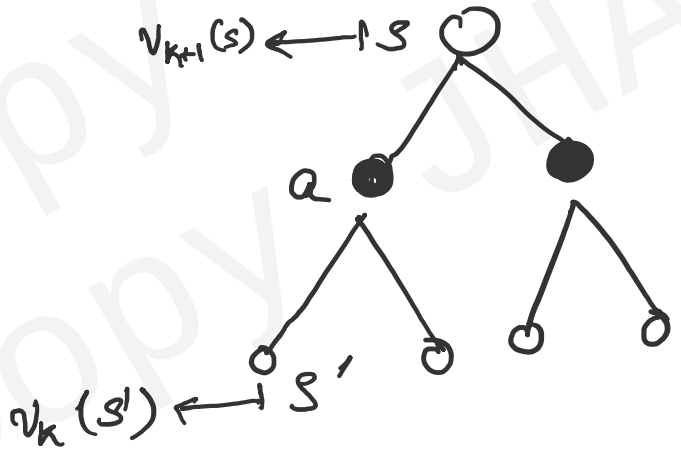
Iterative Policy Evaluation

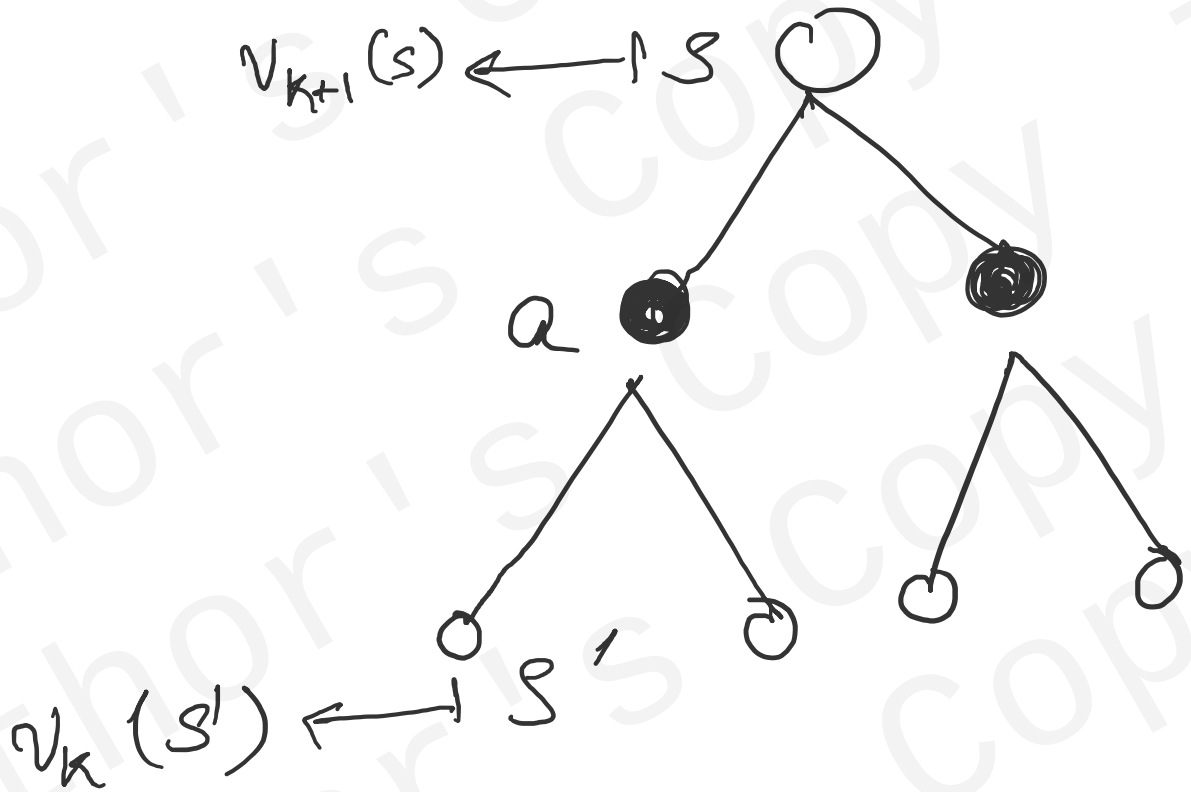
→ Assumes Agent knows MDP, rewards that characterise environment.

Algorithm : to evaluate a given policy π

(iterative application of Bellman expectation Backup)

- $v_1 \rightarrow v_2 \rightarrow \cdots v_\pi$
- Use synchronous back-ups (storage)
  - At iteration k+1
  - For all states $s \in S$
  - Update $v_{k+1}(s)$ from $v_k(s')$





$$V_{k+1}(s) = \sum_{a \in A(s)} \pi(a|s) \sum_{s' \in S, r \in R} p(s', r | s, a)\left(r + \gamma V_k(s')\right)$$

for all $s \in S$.

Iterative Policy Evaluation

Iterative algorithm adapts the Bellman Eq to be used as update rule:

$$V(s) \leftarrow \sum_{a \in A} \pi(a|s) \sum_{s' \in S, r \in R} p(s', r | s, a)(r + \gamma V(s'))$$

Value function

Value function.

Iterative Policy Evaluation

Iterative algorithm

Input: MDP, policy π

Output: V-function (v_π)

$V(s) \leftarrow 0$ for all $s \in S$

repeat:

for $s \in S$

$$V(s) \leftarrow \sum_{a \in A} \pi(a|s) \sum_{s' \in S, r \in R} p(s', r | s, a)(r + \gamma V(s'))$$

Iterative Policy Evaluation

Iterative algorithm

Input: MDP, policy π

Output: V-function (v_π)

$V(s) \leftarrow 0$ for all $s \in S$ | begin with initial guess

repeat:

for $s \in S$ | loop over all states

$$V(s) \leftarrow \sum_{a \in A} \pi(a|s) \sum_{s' \in S, r \in R} p(s', r | s, a)(r + \gamma V(s'))$$

| update the values

If value fn → contraction mapping, state → compact set guarenteed to convery to v_π.

Iterative Policy Evaluation

→ Guaranteed to converge to V_π in infinite sense.

→ But how to assess if convergence is reached in approximate sense?

→ Introduce a ~~θ~~ ~~θ~~ to V_π.

Iterative Policy Evaluation

Input: MDP, policy π, small positive θ

Output: V-function (v_π)

$V(s) \leftarrow 0$ for all $s \in S$ | begin with initial guess

repeat untill $\Delta < \theta$:

$\Delta \leftarrow 0$

for $s \in S$ | loop over all states

$v \leftarrow V(s)$

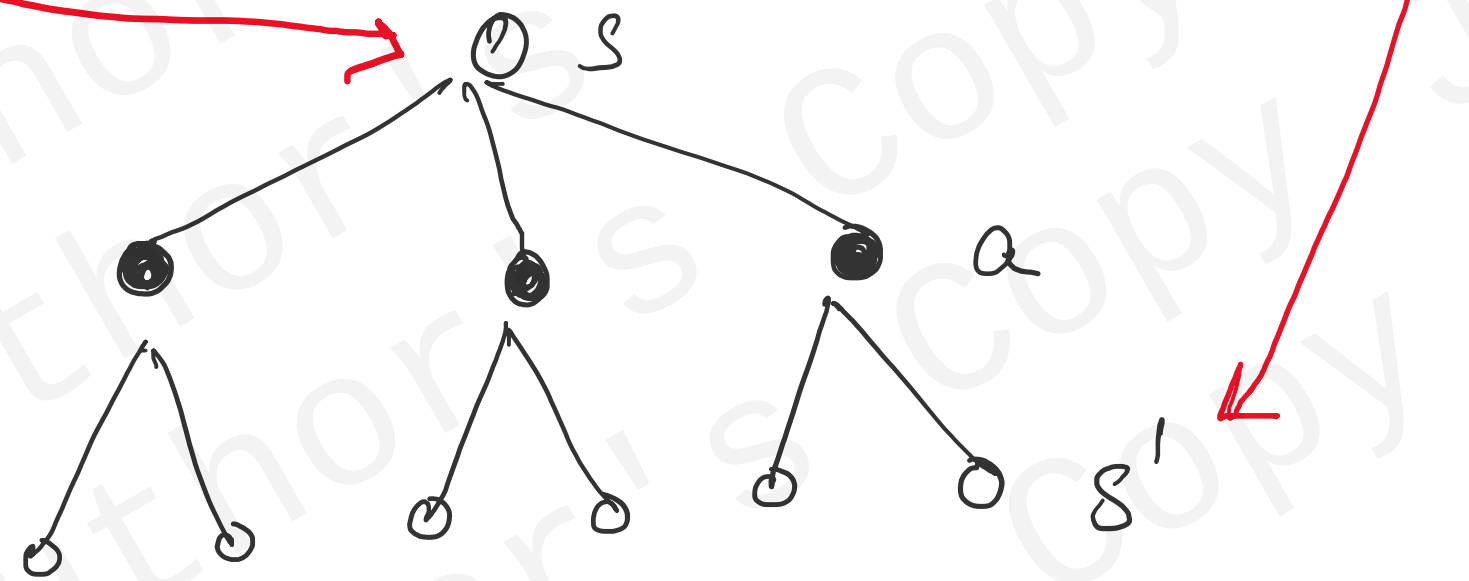
$$V(s) \leftarrow \sum_{a \in A} \pi(a|s) \sum_{s' \in S, r \in R} p(s', r | s, a)(r + \gamma V(s'))$$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$ | how much values of each state changed

if not changed, stop.

return V

$$V(s) \leftarrow \sum_{a \in A} \pi(a|s) \sum_{s' \in S, r \in R} p(s', r | s, a)\left(r + \gamma V(s')\right)$$



Iterative Policy Evaluation

Policy π

↓

Value function v_π

We have v_π

↓

But, optimal v-function still not found !!

Policy Improvement

JHA Mayank , Email: mayank-shekhar.jha [at] univ-lorraine.fr

How to improve a Policy

Policy Evaluation

policy π

↓

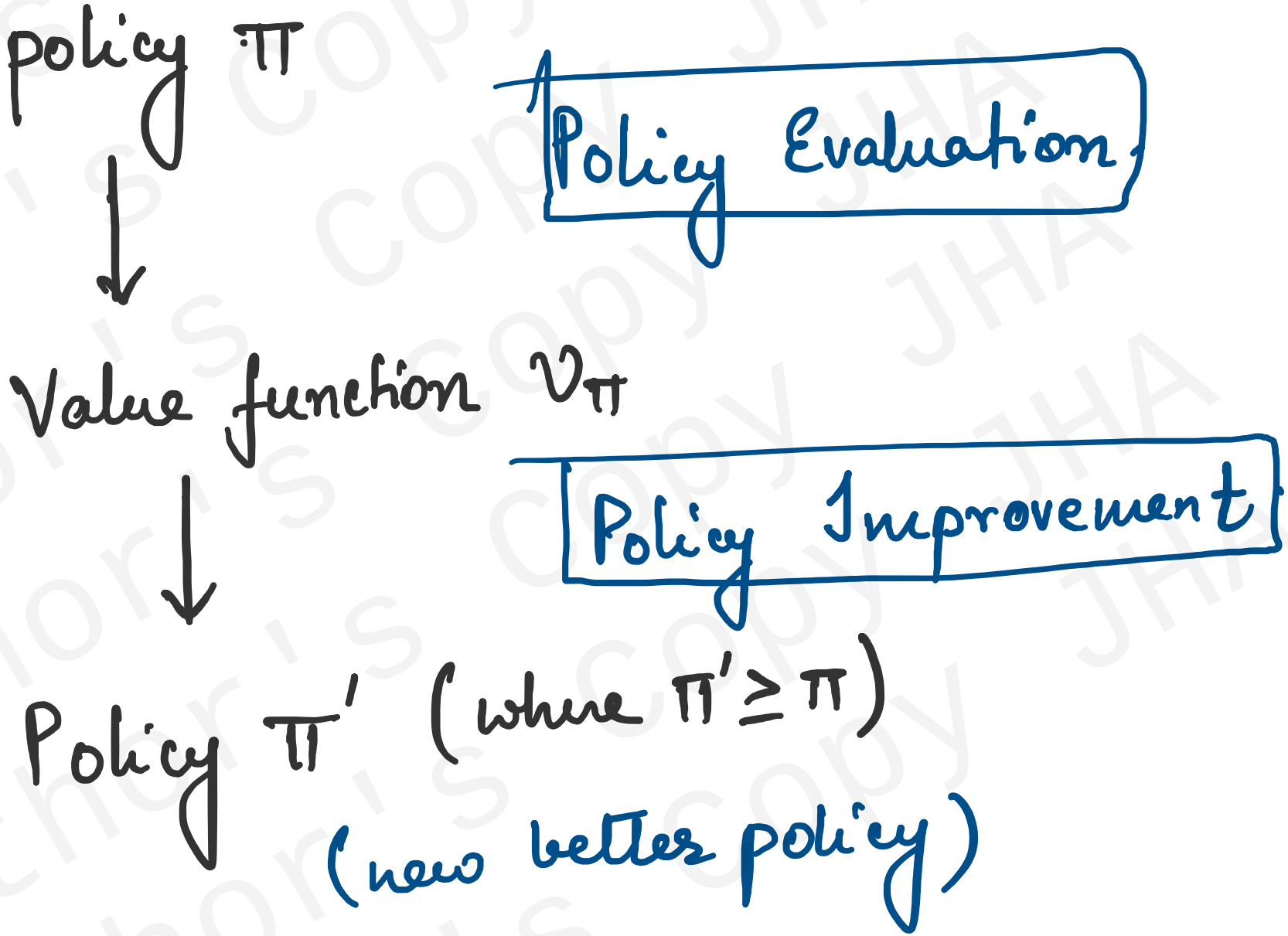
Value function v_π

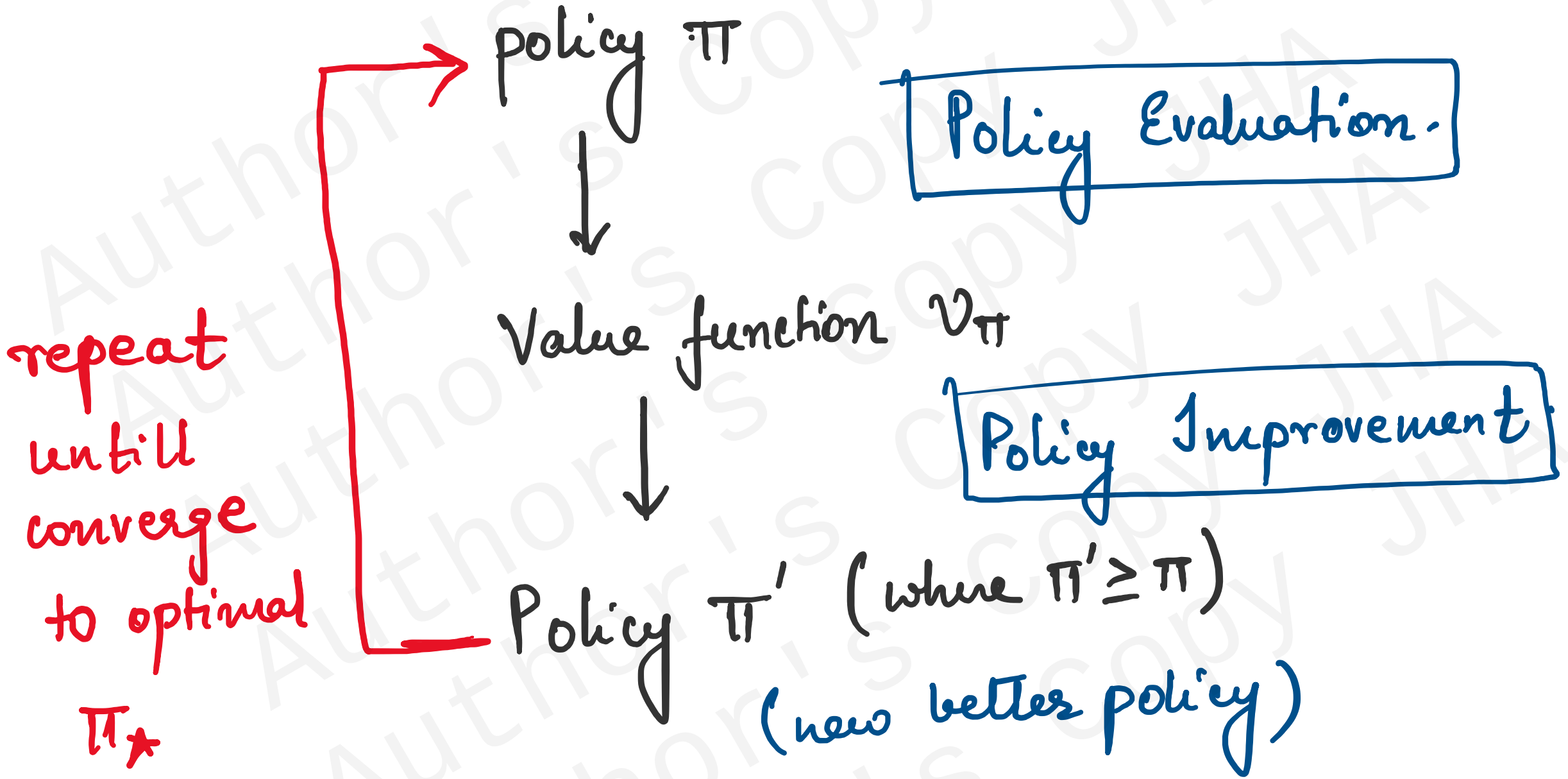
Policy Improvement

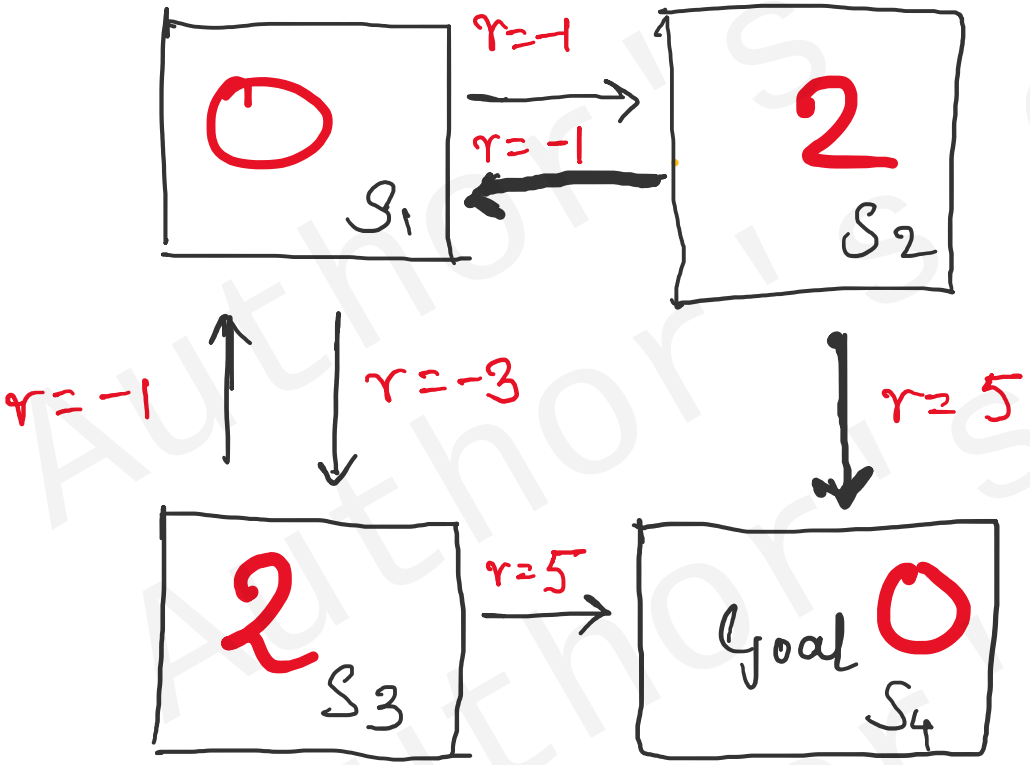
Value function v_π

↓

Policy π' (with $\pi' \geq \pi$)







policy π

Policy Evaluation.

Value function V_π

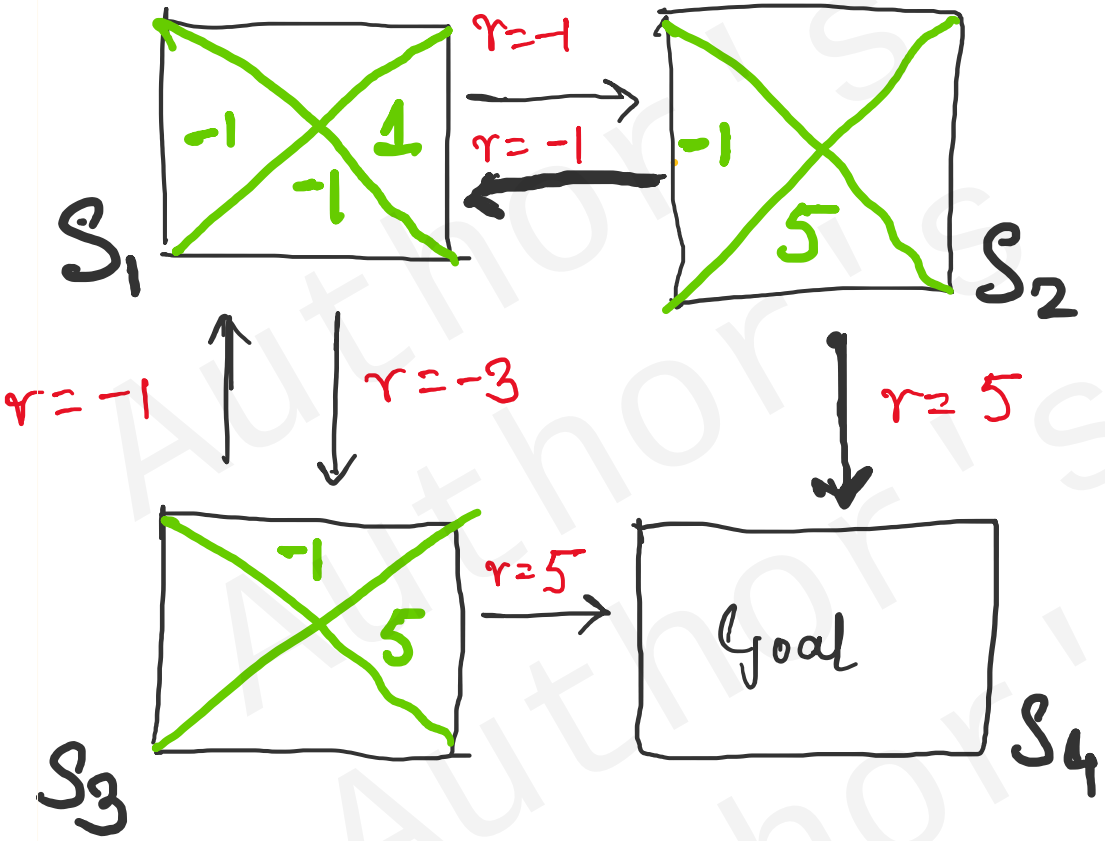
Pliy Improvement.

Policy π' (where $\pi' \geq \pi$)

(new better policy)

Now, how to improve policy.

choose policy $\pi' = greedy(\pi)$ → gives better value for all V_π.



policy π

Policy Evaluation.

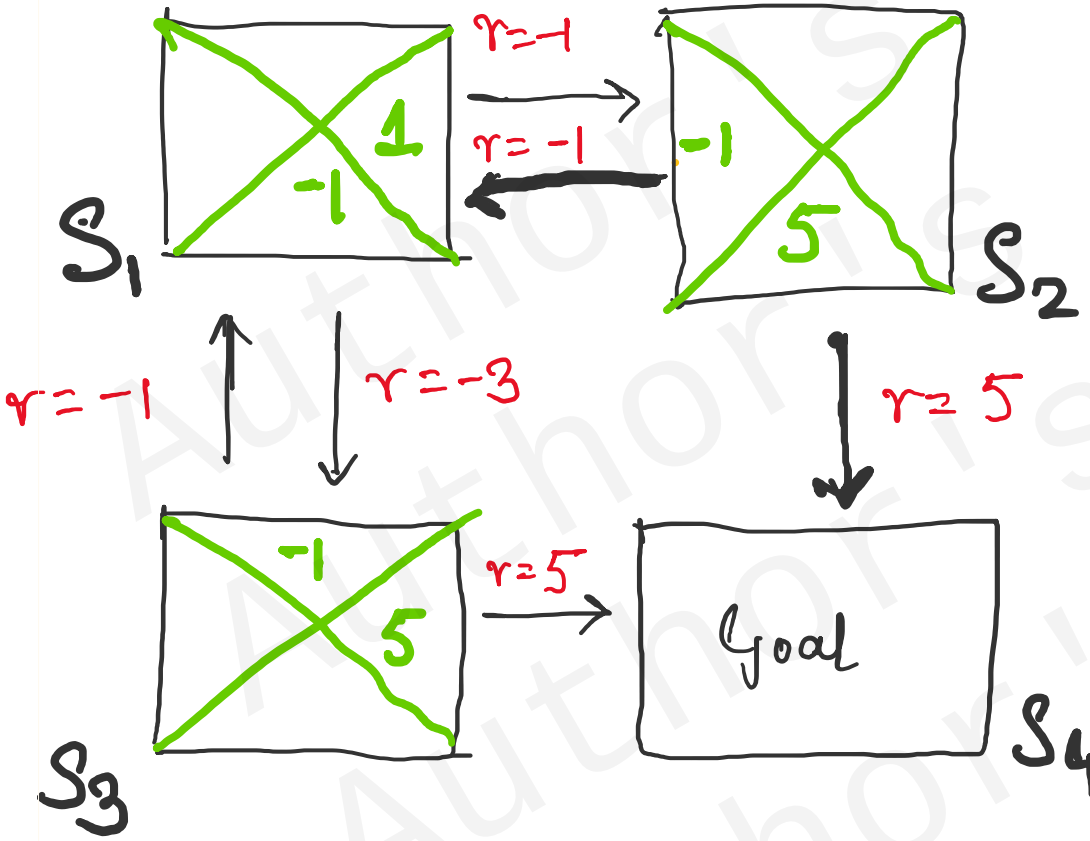
Value function v_π

Action Value Function

(Q-function) q_π

Policy π' (where $\pi' \geq \pi$)

Policy Improvement



policy π

Policy Evaluation.

Value function v_π

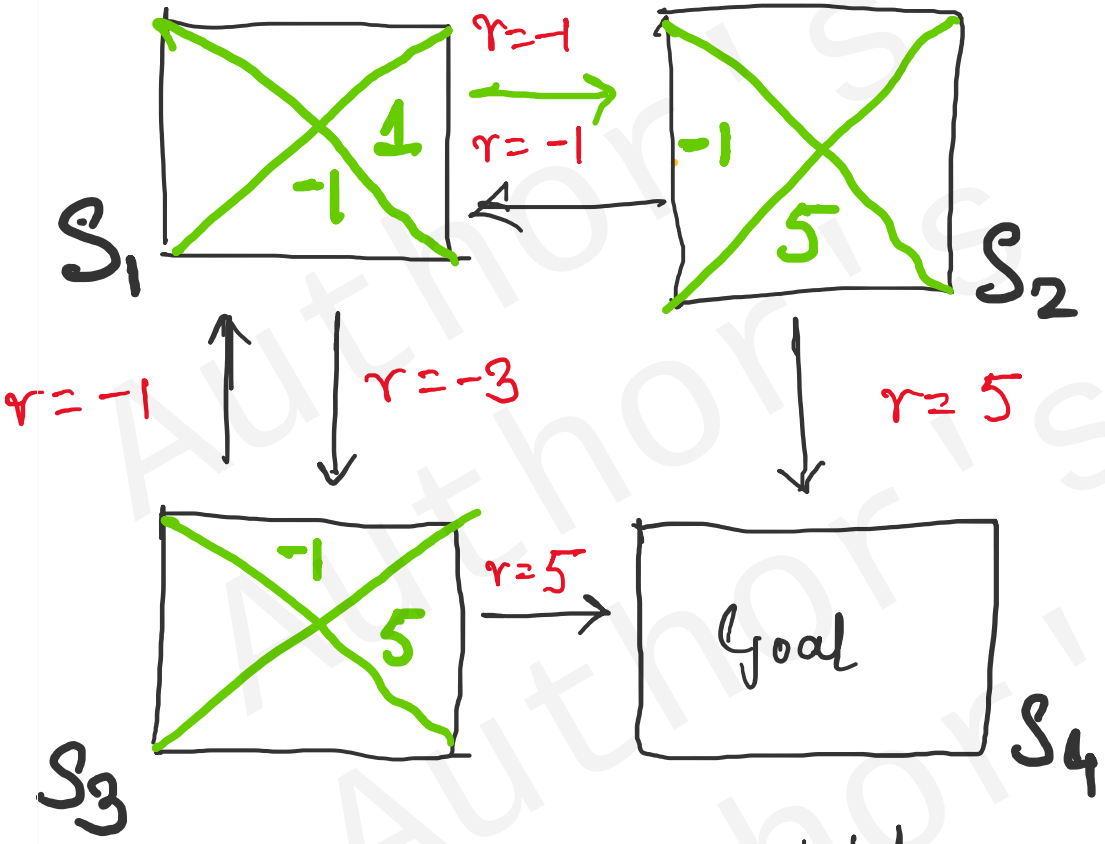
Action Value Function

(Q-function) q_π

Policy π' (where $\pi' \geq \pi$)

Policy Improvement

Focus: How to use q_π to choose better policy π'



When in S_1, go right.

policy π

↓

Value function v_π

↓

Action Value Function

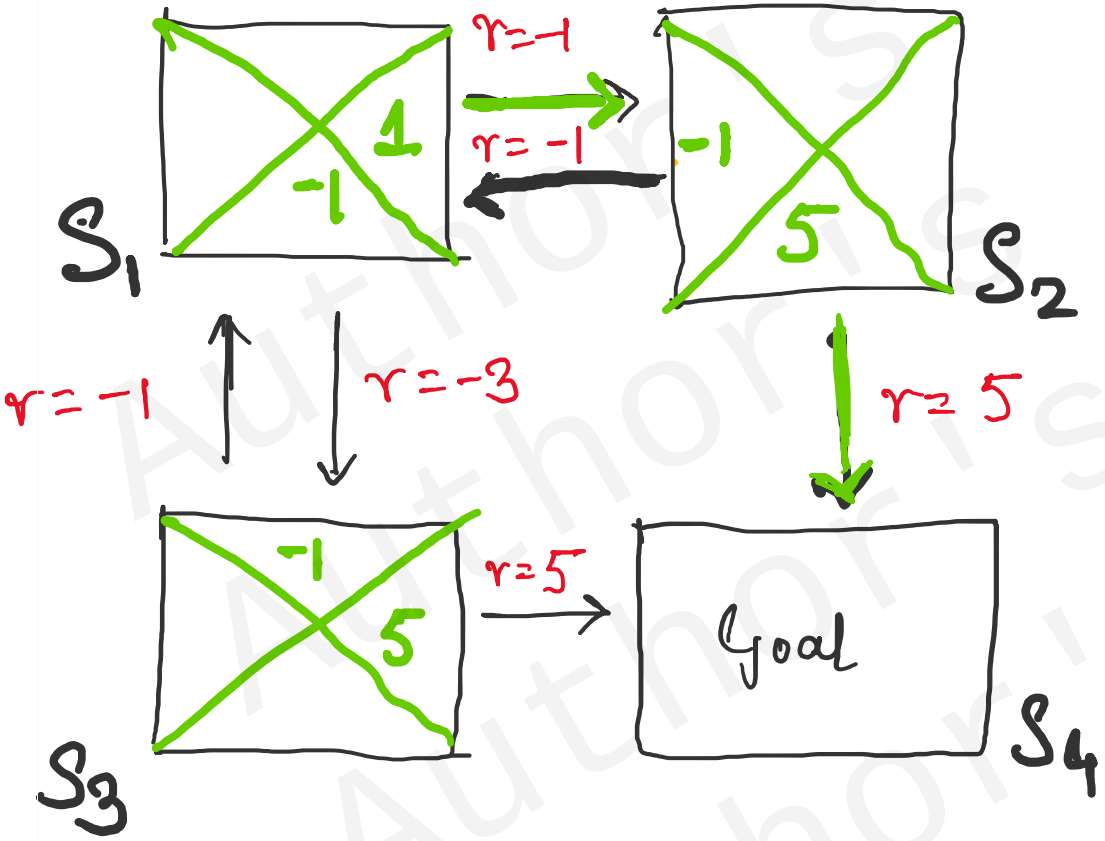
(Q-function) q_π

↓

Policy π' (where $\pi' \geq \pi$)

Policy Evaluation.

Policy Improvement.



When in S_2, go down.

policy π

↓

Value function v_π

↓

Action Value Function

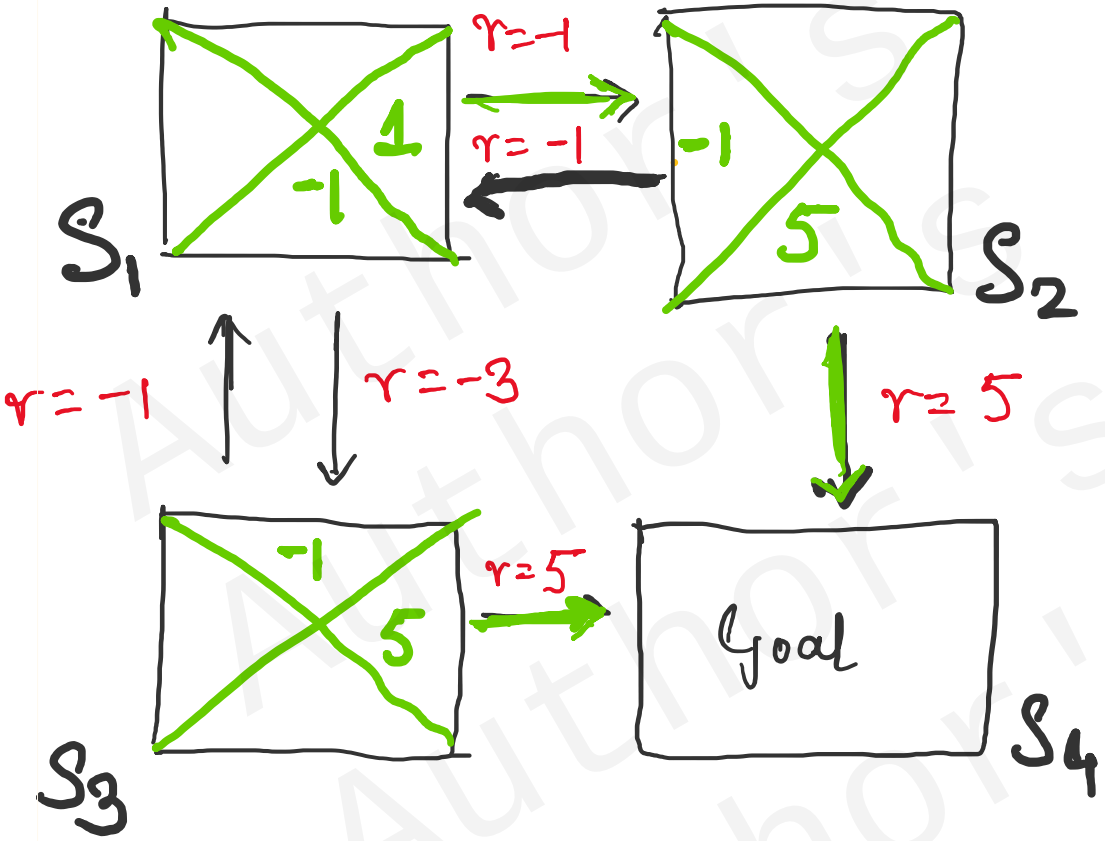
(Q-function) q_π

↓

Policy π' (where $\pi' \geq \pi$)

Policy Evaluation.

Policy Improvement



When in S_3, go right.

Optimal policy in Green. ←

policy π

↓

Value function v_π

↓

Action Value Function

(Q-function) q_π

↓

Policy π' (where $\pi' \geq \pi$)

(if and only if, $v_{\pi'}(s) \geq v_\pi(s)$ for all $s \in S$)

Policy Evaluation.

Policy Improvement.

Policy Improvement

- Consider a policy $a = \pi(s)$
- Improve the policy by <u>greedy</u> approach.

$$\pi'(s) = \underset{a \in A}{\operatorname{argmax}}\, q_\pi(s,a)$$

- This improves value from any state 's' over one step.

$$q_\pi(s, \pi'(s)) = \max_{a \in A} q_\pi(s,a) \geq q_\pi(s, \pi(s)) = v_\pi(s)$$

Policy Improvement:

Value function $V_{\pi'}(s) \geq V_{\pi}(s)$ is improved.

$$V_{\pi}(s) \leq q_{\pi}(s, \pi'(s)) = \mathbb{E}_{\pi'}\left(R_{t+1} + \gamma V_{\pi}(S_{t+1}) \mid S_t = s\right)$$

$$\leq \mathbb{E}_{\pi'}\left(R_{t+1} + \gamma q_{\pi}(S_{t+1}, \pi'(S_{t+1})) \mid S_t = s\right)$$

$$\leq \mathbb{E}_{\pi'}\left(R_{t+1} + \gamma R_{t+2} + \gamma^2 q_{\pi}(S_{t+2}, \pi'(S_{t+2})) \mid S_t = s\right)$$

$$\leq \mathbb{E}_{\pi'}\left(R_{t+1} + \gamma R_{t+2} + \ldots \mid S_t = s\right) = V_{\pi'}(s)$$

Thus, $V_{\pi}(s) \leq V_{\pi'}(s)$

Policy Improvement

→ If improvements stop,

$$q_\pi(s, \pi'(s)) = \max_{a \in A} q_\pi(s,a) = q_\pi(s, \pi(s)) = v_\pi(s)$$

→ Then, Bellman Optimality is stopped;

$$v_\pi(s) = \max_{a \in A} q_\pi(s,a)$$

→ Therefore, $v_\pi(s) = v_*(s)$ for all $s \in S$

π is optimal policy

Policy iteration algorithm

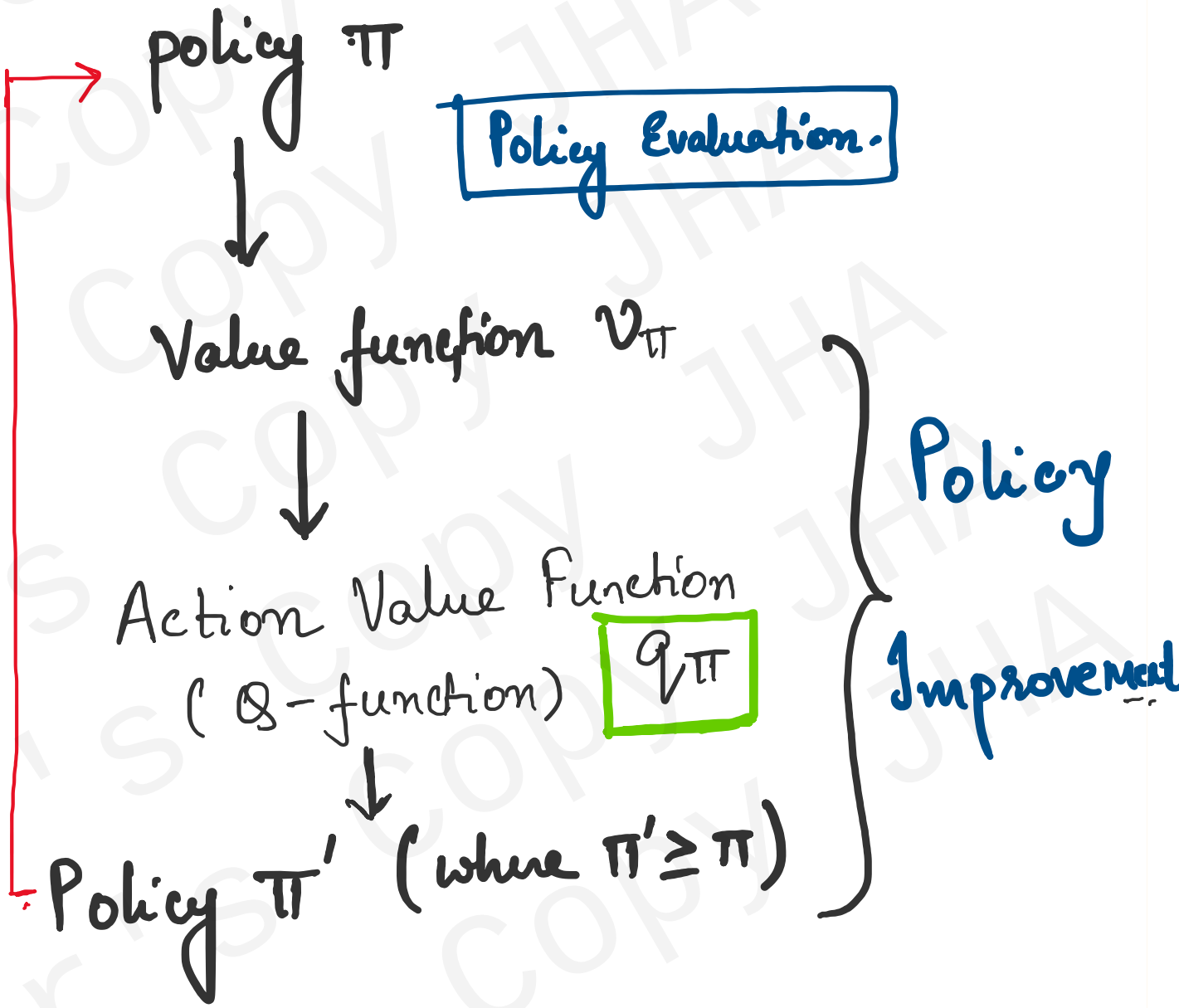
=

Policy evaluation

+

Policy improvement

Policy Iteration



Policy Iteration

=

Policy Evaluation

+

Policy Improvement

for $s \in S$

$$V(s) \leftarrow \sum_{s' \in S, r \in R} p(s', r \mid s, \pi(s)) \left(r + \gamma V(s')\right)$$

for $s \in S$

for $a \in A(s)$

$$Q(s,a) \leftarrow \sum_{s' \in S, r \in R} p(s', r \mid s, a) \left(r + \gamma V(s')\right)$$

$$\pi'(s) \leftarrow \arg\max_{a \in A(s)} Q(s,a)$$

Policy Iteration Algorithm

Input: MDP, small positive number Θ

Output: policy $\pi = \pi_*$

$\pi(a|s) = \frac{1}{|A(s)|}$ for all $s \in S$ and $a \in A(s)$ | begin with equiprobable random policy

repeat:

$V \leftarrow$ Policy-Evaluation (MDP, π, Θ) | obtain V-fun wrt π

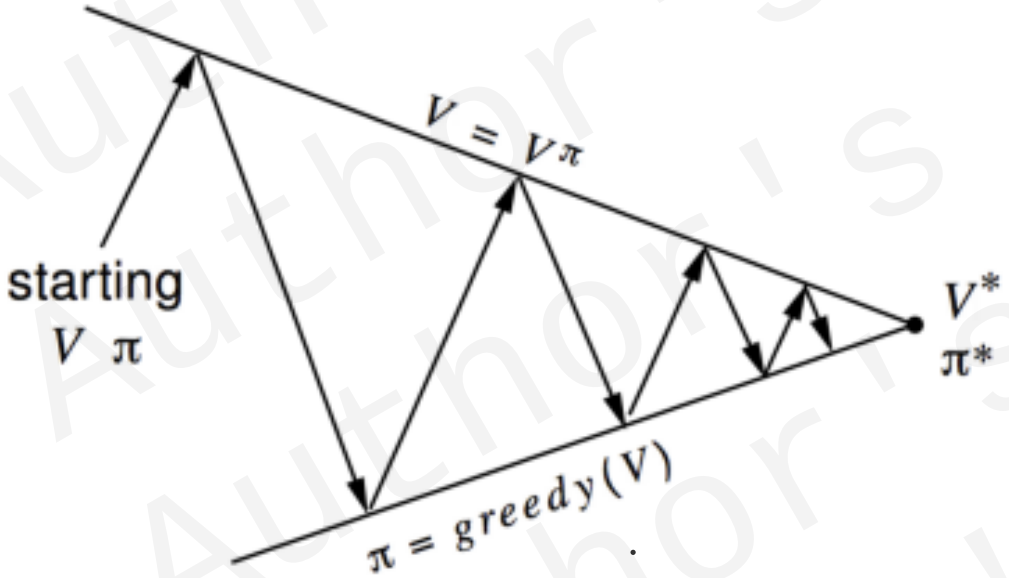
$\pi' \leftarrow$ Policy-Improvement (MDP, V) | obtain better or equal policy from V-function

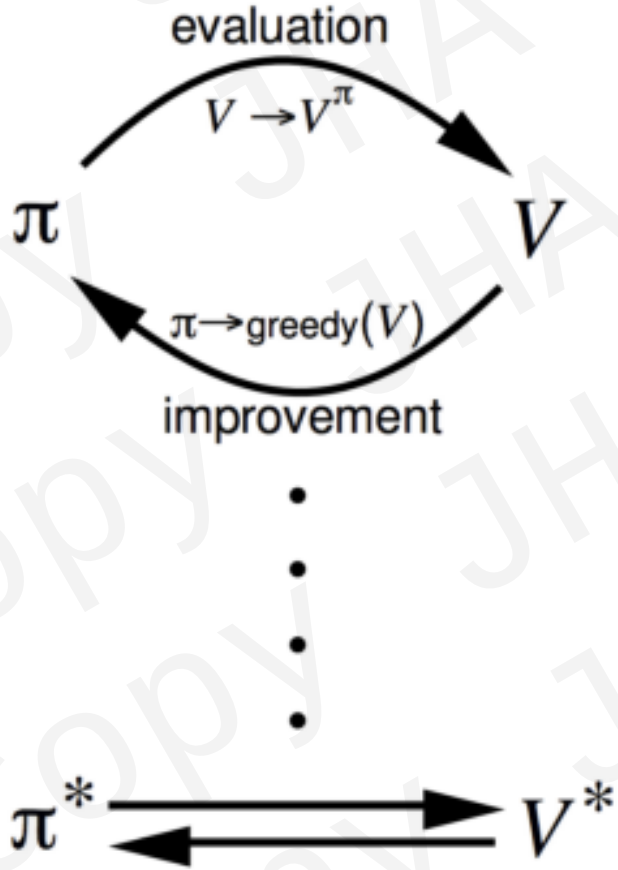
if $\pi = \pi'$ then Break

$\pi \leftarrow \pi'$

return π

Policy Iteration

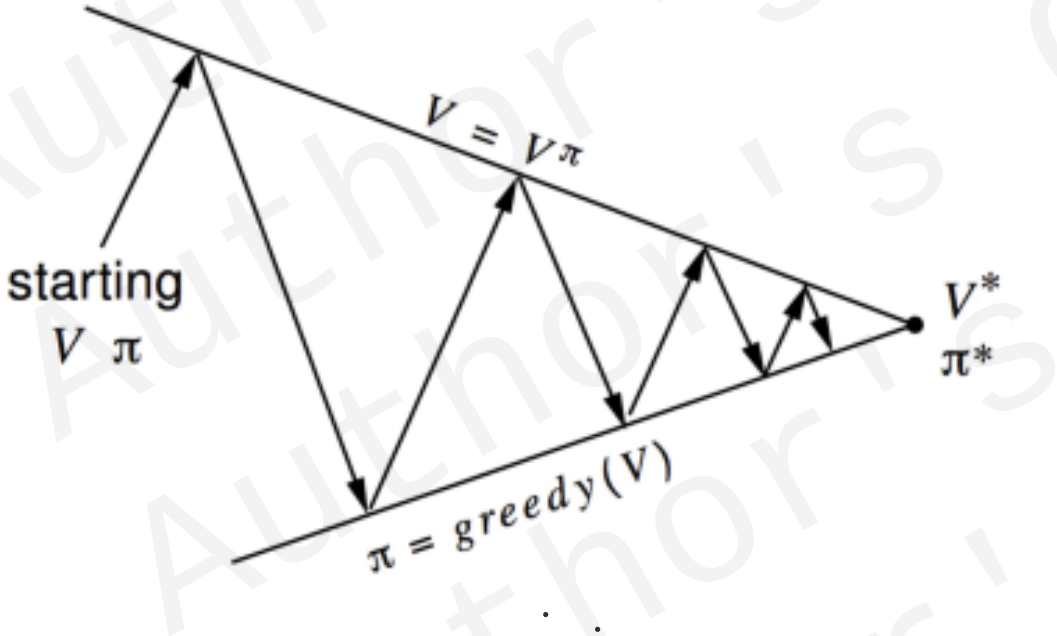


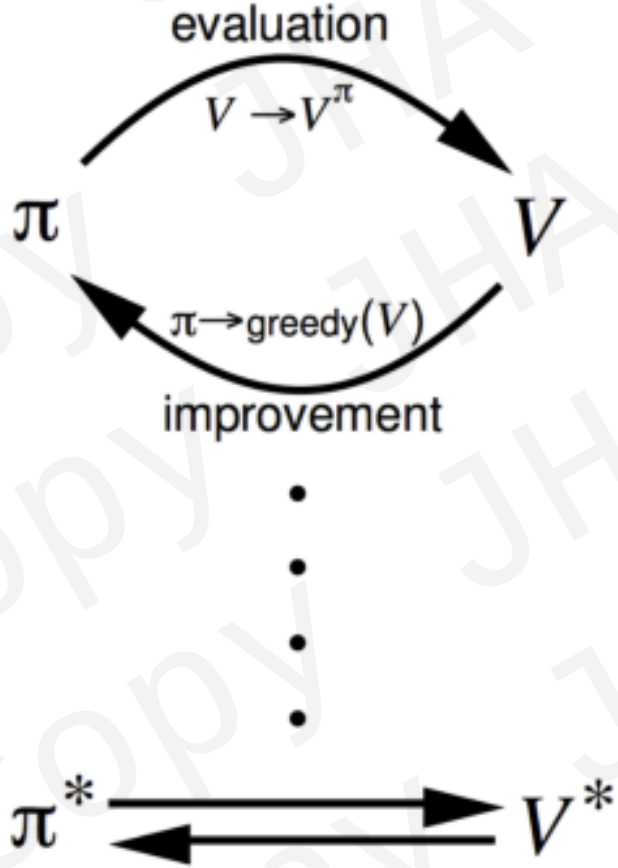


Policy Evaluation : Estimate V_π

Policy Improvement : Greedy ($\pi' \geq \pi$)

Generalised Policy Iteration





Principle of Optimality

Optimal policy = optimal first action A_* + optimal policy from successor state s'.

Theorem: A policy $\pi(a|s)$ achieves optimal value from state s,

$v_\pi(s) = v_*(s)$ if and only if

- any state s' is reachable from s
- π achieves optimal value from s'

  or $v_\pi(s') = v_*(s')$

Value Iteration algorithm

~~Policy Iteration~~ Value Iteration

Policy Evaluation (one time)

for $s \in S$

$$V(s) \leftarrow \sum_{s' \in S, r \in R} p(s', r | s, \pi(s)) \left(r + \gamma V(s')\right)$$

Policy Improvement.

for $s \in S$

for $a \in A(s)$

$$Q(s,a) \leftarrow \sum_{s' \in S, r \in R} p(s', r | s, a) \left(r + \gamma V(s')\right)$$

$$\pi'(s) \leftarrow \arg\max_{a \in A(s)} Q(s,a)$$

~~Policy Iteration~~ Value Iteration

Policy Evaluation (one time)

for $s \in S$

$$V(s) \leftarrow \sum_{s' \in S, r \in R} p(s', r | s, \pi(s)) (r + \gamma V(s'))$$

Policy Improvement.

for $s \in S$

$$\pi'(s) \leftarrow \arg\max_{a \in A(s)} \sum_{s' \in S, r \in R} p(s', r | s, a) (r + \gamma V(s'))$$

~~Policy Iteration~~ Value Iteration

Policy Evaluation (one time)

for $s \in S$

$$V(s) \leftarrow \sum_{s' \in S, r \in R} p(s', r | s, \pi(s)) \left(r + \gamma V(s')\right)$$

Policy Improvement:

for $s \in S$

$$\pi'(s) \leftarrow \arg\max_{a \in A(s)} \sum_{s' \in S, r \in R} p(s', r | s, a) \left(r + \gamma V(s')\right)$$

Policy Evaluation: $\pi \leftarrow \pi'$

(single sweep)

for $s \in S$

$$V(s) \leftarrow \sum_{s' \in S, r \in R} p(s', r | s, \pi(s)) \left(r + \gamma V(s)\right)$$

Redundancy Here

Policy Iteration Value Iteration

Policy Evaluation (one time)

for $s \in S$

$$V(s) \leftarrow \sum_{s' \in S, r \in R} p(s', r | s, \pi(s)) (r + \gamma V(s'))$$

Policy Improvement:

for $s \in S$

$$\pi'(s) \leftarrow \arg\max_{a \in A(s)} \sum_{s' \in S, r \in R} p(s', r | s, a) (r + \gamma V(s'))$$

$\pi \leftarrow \pi'$

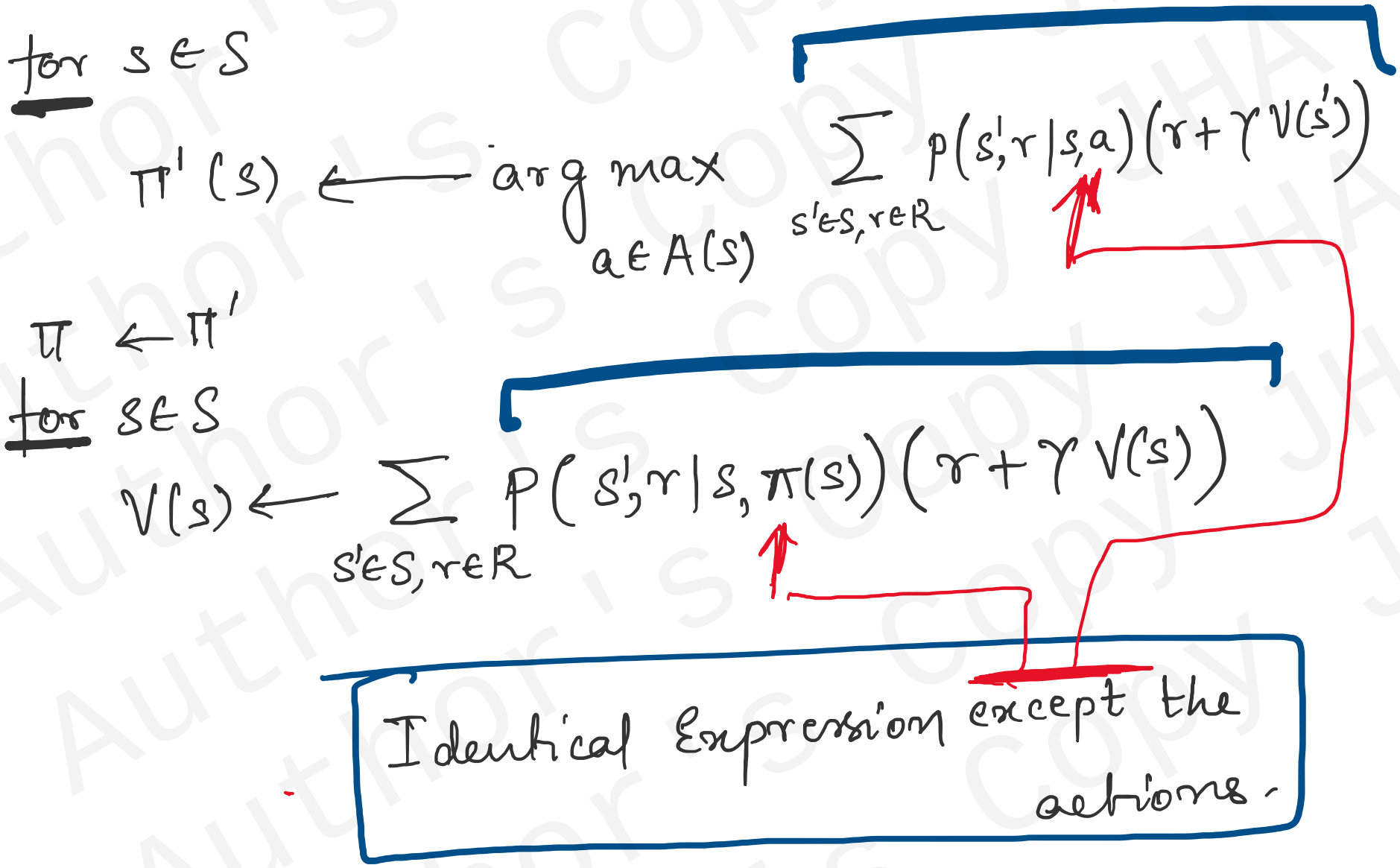
Policy Evaluation:

(single sweep)

for $s \in S$

$$V(s) \leftarrow \sum_{s' \in S, r \in R} p(s', r | s, \pi(s)) (r + \gamma V(s))$$

Redundancy Here


for $s \in S$
$\pi'(s) \leftarrow \arg\max_{a \in A(s)} \sum_{s' \in S, r \in R} p(s', r \mid s, a)(r + \gamma V(s'))$
$\pi \leftarrow \pi'$
for $s \in S$
$V(s) \leftarrow \sum_{s' \in S, r \in R} p(s', r \mid s, \pi(s))(r + \gamma V(s))$
Identical Expression except the actions.

$f(a)$ ①

for $s \in S$

② $\pi(s) \leftarrow \arg\max_{a \in A(s)} \sum_{s' \in S, r \in R} p(s', r \mid s, a)\left(r + \gamma V(s')\right)$

$\pi(s)$ is arg max over 'a'

for $s \in S$

$V(s) \leftarrow \sum_{s' \in S, r \in R} p(s', r \mid s, \pi(s))\left(r + \gamma V(s)\right)$

$f(\pi(s))$ ③

↑ evaluated at $\pi(s)$

Combining ①, ② and ③

(set V(s) to max of the function)

for $s \in S$

$\pi(s) \leftarrow \arg\max_{a \in A(s)} \sum_{s' \in S, r \in R} p(s', r | s, a)(r + \gamma V(s'))$

for $s \in S$

$V(s) \leftarrow \sum_{s' \in S, r \in R} p(s', r | s, \pi(s))(r + \gamma V(s))$

$$V(s) \leftarrow \max_{a \in A(s)} \sum p(s', r | s, a)(r + \gamma V(s'))$$

Value Iteration

Input: MDP, small positive θ

Output: policy π_*

$V(s) \leftarrow 0$ for all $s \in S$

repeat until $\Delta < \theta$.

→ start with some initial guess

$\Delta \leftarrow 0$

for $s \in S$:

→ loop over all states

$v \leftarrow V(s)$

$$V(s) \leftarrow \max_{a \in A(s)} \sum_{s' \in S, r \in R} p(s', r \mid s, a)\,(r + \gamma V(s'))$$

→ update V-fun values

$$\Delta \leftarrow \max(\Delta, |v - V(s)|)$$

→ stop when not updating

for $s \in S$

$$\pi(s) = \arg\max_{a \in A(s)} \sum_{s' \in S, r \in R} p(s', r \mid s, a)\,(r + \gamma V(s'))$$

→ Obtain policy with respect to final value function.

Real Time Dynamic Programming.

1) Use states relevent to the agent

2) Use agent's experience to guide the selection of states

3) After each time step: S_t, A_t, R_{t+1}

4) Back up the state S_t

$$V(S_t) \leftarrow \max_{a \in A(s)} \sum_{s' \in S, r \in R} p(s', r \mid S_t, a)\,(r + \gamma V(s'))$$

Here s' is the state at time $t+1$ or $s' = S_{t+1}$

Back Ups.

1) DP uses full back ups
2) Every successor state is considered
3) knowledge of MDP and reward transition is used
4) For large problems, large state space
   'curse of dimensionality'
   number of states $n = |S|$
   ↓
   grows exponentially with number of variables.